AMENDMENTS TO THE SPECIFICATION

Please amend page 1 of the specification by changing the title of the invention and removing the underlining from the headings as follows:

~~A SHARED RESOURCE QUEUE FOR SIMULTANEOUS MULTITHREADED PROCESSING~~

SHARED RESOURCE QUEUE FOR SIMULTANEOUS MULTITHREADING PROCESSING WHEREIN ENTRIES ALLOCATED TO DIFFERENT THREADS ARE CAPABLE OF BEING INTERSPERSED AMONG EACH OTHER AND A HEAD POINTER FOR ONE THREAD IS CAPABLE OF WRAPPING AROUND ITS OWN TAIL IN ORDER TO ACCESS A FREE ENTRY

~~TECHNICAL FIELD~~ TECHNICAL FIELD

The present invention relates in general to an improved data processor architecture and in particular to a queue within the processor core to support simultaneous hardware multithreading.

~~BACKGROUND OF THE INVENTION~~ BACKGROUND OF THE INVENTION

From the standpoint of the computer's hardware, most systems operate in fundamentally the same manner. Computer processors actually perform very simple operations quickly, such as arithmetic, logical comparisons, and movement of data from one location to another. What is perceived by the user as a new or improved capability of a computer system, however, may actually be the machine performing the same simple operations at very high speeds. Continuing improvements to computer systems require that these processor systems be made ever faster.

Please amend the paragraph beginning on page 7, line 21 through page 8, line 4 as follow:

With respect to the threads there may be private or separate resources, or shared resources. Private resources simplify the management in that pointers from different threads address different data structures and may permit advantageous placement of the queues in different parts of the chip. Some private resources are registers or queues dedicated for the exclusive use of a particular thread. Another example of a private resource may be a split queue or registers having reserved spaces for each thread. An example of a partitioned queue is set forth in U.S. Patent Application Serial No. ~~09/645,08~~ 09/645,081 filed 24 August 2000 entitled *Method for Implementing a Variable-Partitioned Queue for Simultaneous Multithreaded Processors*, which application is owned by the assignee herein and which is hereby incorporated by reference in its entirety.

On page 9, lines 7 through 14, please replace the following heading and paragraph as follows:

~~SUMMARY OF THE INVENTION~~ SUMMARY OF THE INVENTION

These needs and others that will become apparent to one skilled in the art are satisfied by a resource queue, comprising: a plurality of entries, each entry having unique resources required for information processing in which the plurality of entries is allocated amongst a plurality of independent hardware threads such that the resources of more than one thread may be within the queue and the entries allocated to one thread may be interspersed among the entries allocated to another thread.

Please amend the paragraph on page 15, lines 4-21 as follows:

CPU 126 is a general-purpose programmable multithreaded processor, executing instructions stored in memory 158. While a single CPU having multithreaded capabilities is shown in FIG. 1, it should be understood that computer systems having multiple CPUs, some of which may not have multithreaded capabilities, are common in servers and can be used in accordance with principles of the invention so long as one CPU has multithreading capabilities. Although the other various components of FIG. 1 are drawn as single entities, it is also more common that each consist of a plurality of entities and exist at multiple levels. While any appropriate multithreaded processor can be utilized for CPU 126, it is preferably one of the PowerPC™ line of microprocessors available from IBM having simultaneous multithreading capabilities. Processing unit 112 with CPU 126 may be implemented in a computer, such as an IBM pSeries or an IBM iSeries computer running the AIX or other operating system. CPU 126 accesses data and instructions from and stores data to volatile random access memory (RAM) 158. CPU 126 is suitably programmed to carry out the preferred embodiment as described in more detail in the flowcharts of FIGS. 5-11B.

On page 17, line 13 through page 18, 2, please amend the paragraph as follows:

Shown in FIG. 2 is a multithreaded computer processor architecture 210 in accordance with a preferred implementation of the invention. Illustrated is an out-of-order pipelined processor such as that disclosed in *System and Method for Dispatching Groups of Instructions*, U.S. Serial No. 09/108,160 filed 30 June 1998; *System and Method for Permitting Out-of-Order Execution of Load Instructions*, U.S. Serial No. 09/213,323 filed 16 December 1998; *System and Method for Permitting Out-of-Order Execution of Load and Store Instructions*, U.S. Serial No. 09/213,331 filed 16 December 1998; *Method and System for Restoring a Processor State Within a Data Processing System in which Instructions are Tracked in Groups*, U.S. Serial No. 09/332,413 filed 14 July 1999; *System and Method for Managing the Execution of Instruction Groups Having Multiple Executable Instructions*, U.S. Serial No. 09/434,095 filed 05 November 1999; *Selective Flush of Shared and Other Pipelined Stages in a Multithreaded Processor*, U.S. Serial No. 09/564,930 filed 04 May 2000; and *Method for Implementing a Variable-Partitioned Queue for Simultaneous Multithreaded Processors*, U.S. Serial No. ~~09/645,08~~ 09,645,081 filed 24 August 2000, all these patent applications being commonly owned by the assignee herein and which are hereby incorporated by reference in their entireties.

On page 21, line 22 through page 22, line 28, please substitute the following two paragraphs:

As discussed, though, separate data structures and queues are not as efficient as a structure shared amongst the threads because either the structure is fixed or changes so slowly to be unable to accommodate dynamic and responsive processing. Thus, the preferred embodiment of the invention further contemplates that the data structures and registers be architected as shared resources. Shared resources are those processor registers and queues which can be shared by either thread either separately or at the same time. In some circumstances, the non-renamed tracking register 242, the LRQ 244, the SRQ 246, the GCT 248, the register renamed pools 250, the issue queues 222, the branch information queue (BIQ) ~~(not shown)~~ 252 may be shared amongst threads. Alternatively, some resources may be shared while others may be private, e.g., the GCT 248 may be shared while the LRQ 244 and the SRQ 246 may be private depending upon the particular architecture. The use of shared resources for the various queues yields performance benefits without increasing the chip area over the shared queue case.

FIG. 3 is a diagram of a shared queue 300 in accordance with an embodiment of the invention. This queue 300 may be any of the queues which are shared by the multiple threads, such as a BIQ ~~(not shown)~~ 252 and other queues shown in FIG. 2, e.g., the LRQ 244, the SRQ 246, and the GCT 248. The shared queue 300 has a number of entries 310 - 332. Each entry has a number of fields, 340 - 346; field 340 identifies the thread; field 342 holds a bank number which represents how many times a head pointer has passed a tail pointer for a particular thread, as will be explained; valid/invalid field 344 indicates if the entry contains valid data; and content-specific field 346 may hold other content specific to the queue, thread, or entry. For instance, if the queue 300 is used as a GCT, then the content 346 might be, e.g., pointers to the architected and physical registers used by the instructions, the type of instruction, what execution unit would be used, bits indicating when each instruction is a group is

complete, etc. When the shared queue is a LRQ or an SRQ, the queue might contain the virtual addresses of the load/store memory location and/or a pointed to the group in the GCT. The queue 300 of FIG. 3 is illustrated with two independent threads of execution within the pipeline but the concepts apply equally to multiple independent threads greater than two.

On page 35, please remove the underlining from the heading as follows:

~~CLAIMS~~ CLAIMS